INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

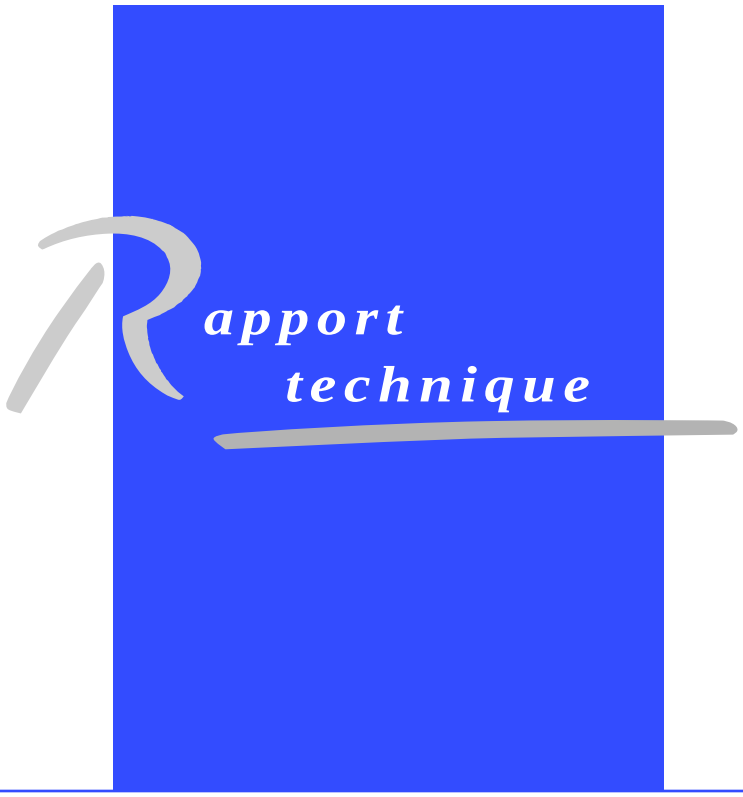
Modélisation de la Main pour sa Localisation dans une Séquence d'Images

Quentin DELAMARRE

N° 0198

Décembre 1996

THÈME 3



ISSN 0249-0803

Modélisation de la Main pour sa Localisation dans une Séquence d'Images

Quentin DELAMARRE *

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet ROBOTVIS

Rapport technique n° 0198 — Décembre 1996 — 45 pages

Résumé : Détecter et estimer la position d'une main dans une séquence d'images demande une attention particulière quant à la méthodologie à employer. C'est un problème difficile, qui se décompose en plusieurs étapes: trouver un modèle de main approprié, détecter les contours dans les images, établir des correspondances entre le modèle et ces contours, en déduire la position dans l'espace de la main de l'utilisateur, pour enfin prédire la position la plus probable de la main dans les images suivantes.
Nous utilisons les contours actifs géodésiques pour trouver le contour de la main dans l'image, mais le fait que la main soit un objet articulé rend complexe son étude. C'est pourquoi nous nous limiterons, dans le cadre de ce stage, à la reconnaissance d'une main plane dans une image.

Mots-clé : localisation d'objets dans une image, modélisation, objets articulés, détection de contours, contours actifs géodésiques.

(Abstract: pto)

Ce rapport technique est le rapport de stage de DEA de Quentin Delamarre.

* . Email : Quentin.Delamarre@sophia.inria.fr

Unité de recherche INRIA Sophia Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA ANTIPOLIS Cedex (France)
Téléphone : 04 93 65 77 77 – Télécopie : 04 93 65 77 65

Modelisation of the Hand for its Tracking in a Sequences of Images

Abstract: To detect and to estimate the pose of a hand in a sequence of images is a difficult task which demands particular attention. This problem can be split into several stages: find an appropriate model for the hand, detect the contour of the hand in the images, establish correspondences between the model and this contour, estimate the pose of the hand, and then predict the pose in the next image.
In order to extract the contour of the hand in the images, we use geodesic active contours.
Since the hand is an articulated object, with a high number of DOF (degrees of freedom), we studied only a planar hand in one image.

Key-words: tracking, model matching, articulated objects, edge detection, geodesics, modelisation.

Table des Matières

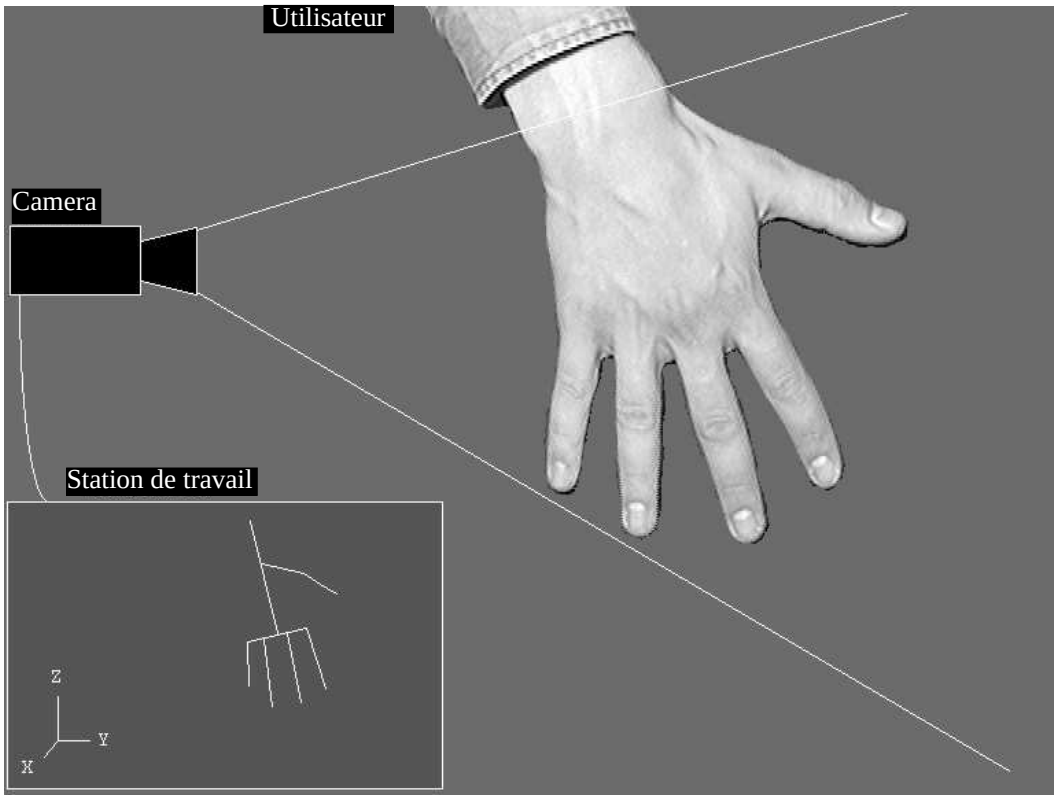


Figure 1 – *Principe général: une main est filmée par une camera video, puis une station de travail estime la position de cette main par rapport à la caméra*

Introduction

Le but de ce stage était d'estimer grâce à un ordinateur la position dans l'espace d'une main filmée par une ou plusieurs caméras vidéo (voir figure 1).

Une ou plusieurs caméras fournissent à la station de travail une séquence d'images de la main de l'utilisateur. Pour en déduire la position relative de cette main par rapport aux caméras, on suppose que l'on connaît la forme générale d'une main (le modèle 3D), et on va essayer de calquer ce modèle sur l'image de la main de l'utilisateur dans la séquence d'images.

De nombreuses études ont déjà été effectuées dans le domaine, mais uniquement pour estimer la position dans l'espace d'objets rigides (comme des pièces usinées qui défilent sur un tapis roulant). La difficulté, ici, est que l'objet à reconnaître n'est pas rigide, mais est articulé.

Utiliser la vision par ordinateur permet d'éviter à l'utilisateur de mettre un gant recouvert de capteurs de position, donc fragile. Cela permet aussi de travailler sur des séquences d'images prises à l'avance; par exemple, pour reproduire le mouvement de la main d'un grand chef d'orchestre à partir du film d'un de ses concerts (serait-il d'accord?) !

Les applications d'un tel système sont nombreuses: reconnaissance automatique et transcription en mots du langage des sourds-muets, interface homme-machine améliorée ("souris 3D"), manipulation d'objets à distance, etc. On peut imaginer étendre ce système à d'autres objets connus (piétons, bras manipulateurs,…).

Une "souris 3D" qui permettrait de prendre avec la main des icônes, ou des fichiers, pour les déplacer ou les copier, aurait beaucoup de succès, vu le nombre croissant de stations et de PCs vendus avec une caméra.

Je vais exposer ce qui existe déjà dans ce domaine, puis ce que j'ai réalisé pendant mon stage, pour finalement décrire les améliorations futures possibles.

1 Etat de l'art

Ce stage s'inscrit dans la reconnaissance d'objets basée sur leur modélisation. Ce résumé des recherches récentes dans ce domaine est inspiré d'un rapport de recherche de A. R. Pope ([POP94]).

1.1 Introduction

Le problème est le suivant: étant données des informations sur des objets (modèle, apparence), et une séquence d'images où ils peuvent apparaître, dire si oui ou non ces objets sont présents, et estimer leur position.

Plusieurs approches sont possibles, qui peuvent être classées de la façon suivante:

– Suivant le type de données que l'on possède sur les objets: leurs modèles, où l'on peut les trouver, ou quelles sont leurs fonctions.

– Suivant le type d'objets à reconnaître: forme 2D, volume simple, ou complexe, articulé ou non, …

– Suivant le type d'images en notre possession: images vidéo monoculaires ou binoculaires, images proximétriques, …

La reconnaissance s'effectue en établissant des correspondances entre des caractéristiques de l'image et nos modèles. Il faut donc trouver quelles sont ces caractéristiques, comment les extraire de l'image, et comment les lier au modèle. Nous nous limiterons ici à des caractéristiques locales (opposées aux caractéristiques globales telles que mesures de surfaces, de longueurs de périmètres, …qui supportent mal les occultations).

Voici quelques critères (cf [GRI90]) pour juger de **l'efficacité d'une méthode**:

– *étendue:* quels types d'objets peuvent être reconnus, dans quels genres de scènes?

– *robustesse:* la méthode supporte-t-elle bien le bruit et les occultations?

– *efficacité:* la gestion de tous les cas de figures et de toutes les possibilités de postures des objets se fait-elle au dépend de la rapidité d'exécution et de la place mémoire prise?

– *exactitude:* le système de reconnaissance d'objets est souvent amené à faire des choix parmi des hypothèses. La méthode fait-elle toujours les bons choix?

Le chapitre suivant discutera des différentes manières de représenter les objets et les images. Après, nous étudierons comment établir des correspondances entre les caractéristiques de l'images et du modèle. Puis nous conclurons sur cet état de l'art.

1.2 Représentation

Pour faciliter les mises en correspondances entre l'image et le modèle, il est utile de rapprocher les types de caractéristiques extraites des deux. Par exemple, si le modèle est décrit par un parallélépipède, il sera plus efficace de chercher des contours d'intensité droits dans l'image.

Il faut se rappeler que le modèle est une description de l'objet, alors que l'image ne contient que son apparence.

Dans la suite de ce chapitre, nous appellerons *forme* (niveau physique) , l'ensemble des points sur la surface d'un objet, sur un contour ou dans une région de l'image, et *représentation* (niveau logique) de la forme, le langage pour décrire cet ensemble (ligne, courbe, surface, …).

Une représentation d'une forme est donc la description des parties de la forme, en explicitant comment chacune d'elles est liée à la forme totale.

Pour clarifier cela, voici deux exemples:
- Dans le cas d'un contour d'un objet dans une image, la forme du contour est la liste des points de cette courbe. La représentation de la forme est la longueur de la courbe, la liste des segments reliant chacun des points, la valeur de la courbure le long du contour, …
- Dans le cas d'un modèle 3D, la forme est l'ensemble des points à la surface de ce modèle. La représentation est la liste des facettes dans l'espace, les coordonnées des sommets dans un repère, quelques paramètres (l'angle entre deux parties d'un modèle articulé par exemple), …

Avoir une bonne représentation d'une forme:

Voici quelques critères pour décider de **l'efficacité d'une représentation** aussi bien du modèle que des formes dans l'image:

- *étendue et sensibilité:* la représentation doit pouvoir décrire toutes les formes nécessaires, tout en permettant de les distinguer.
- *unicité:* chaque forme ne devrait avoir qu'une seule description pour simplifier la comparaison des formes.
- *stabilité:* un faible changement de la forme ne doit produire qu'un faible changement dans la description. Ceci simplifie aussi la comparaison des formes.
- *efficacité:* il doit être possible de calculer efficacement une description de formes aussi bien pour le modèle que dans l'image. Il doit être facile de comparer des description de formes.

Cette liste conduit à quelques conséquences:
D'abord une forme devrait être décrite comme un assemblage de primitives locales simples. En effet, dans ce cas chacune d'elles (des segments par exemple) sera plus facilement déductible de l'image ou du modèle. Qui plus est, la description sera plus

stable puisqu'une partie seulement des primitives sera affectée par un petit déplacement de l'objet dans l'image, ou par l'apparition d'une occultation.

Une représentation découpe une forme en primitives selon trois axes de discrétisation: l'emplacement (au pixel près), l'échelle, et la catégorie (par exemple: "est-ce un plan ou une portion de cylindre?"). Il n'est pas très grâve que cela aille à l'encontre des critères d'unicité et de stabilité.

En fait, le choix des primitives dépend de la nature de l'application.

Le choix du système de coordonnées:

On distingue en général deux façons de décrire *une représentation de formes*:

Représentation centrée objet:

On décrit **l'objet** à reconnaître:

- soit en attachant un repère à cet objet et en donnant la liste des coordonnées des points de sa surface,
- soit en attachant un repère à chacune de ses parties (par exemple pour un objet articulé), et en explicitant leur position dans une hiérarchie de l'objet (comme dans [BRO81, REH95, LEE95, HOG84]).

Quant à **la forme à reconnaître dans l'image**, là encore plusieurs méthodes existent:

- on peut là aussi imaginer une description centrée des formes dans l'image. Toutefois cette méthode supporte mal les occultations, et il est difficile de déduire des formes de l'image.
- ou alors on compare l'image avec la projection de l'objet 3D dans celle-ci. Cette méthode fait donc intervenir la perspective et la gestion des occultations, ce qui complexifie l'algorithme.

Représentation centrée vue:

L'alternative ici est de représenter l'objet par l'ensemble de ses apparences sous tous les points de vue possibles. Il ne reste donc plus qu'a rechercher quelle apparence de l'objet se retrouve dans l'image. L'inconvénient de cette méthode est la place mémoire requise pour enregistrer toutes les combinaisons. Toutefois des algorithmes existent pour réduire cet espace (cf [REH95]).

Combiner les deux types de représentation:
Il est possible d'utiliser la représentation centrée-objet pour déceler rapidement un objet dans une image, puis d'utiliser la représentation centrée-vue pour vérifier la détection.

1.2.1 Invariants

Certaine primitives sont indépendantes du point de vue sur l'objet. Les invariants géométriques sont des grandeurs qui restent constantes sous certains types de transformations.

Un exemple est le bi-rapport qui est un invariant projectif. Quels que soient quatre points A, B, C et D collinéaires, la valeur (AC.BD)/(AD.BC) est la même mesurée sur l'objet réel et sur sa projection dans l'image. De nombreuses études ont été suivies pour extraire ce genre de points dans une image (droites parallèles, …). Toutefois il a été prouvé qu'il n'existait pas d'invariant projectif pour une combinaison quelconque de points.

Le fait qu'un objet cylindrique possède un contour à peu près plan sous tous les points de vue, ou que le contour d'intensité d'un objet varie très peu avec l'éclairage sont d'autres types d'invariants utiles.

1.2.2 Primitives de forme usuelles

Quels sont les différents **types de primitives** que l'on peut utiliser dans la reconnaissance d'objets?

Segments et facettes: Beaucoup de systèmes utilisent les segments pour approximer des courbes. Toutefois cela ne convient pas pour certaines formes (elliptiques par exemple) qui sont alors décomposées en une multitude de segments. C'est pourquoi il peut être utile d'utiliser des courbes d'ordre supérieur (comme les arcs d'ellipses, les coniques généralisées, les splines 3D, etc.), qui réduisent le nombre de primitives nécessaires, parfois aux dépends de la stabilité.

Parties: Un objet peut être subdivisé en éléments simples tels que des cylindres généralisés (voir [BRO81]), des super-quadriques, ou même des "blobs", sortes d'objets à densité de probabilité (voir [PEN95]).

Caractéristiques distinctes: Dans ce cas, on ne cherche qu'à décrire les zones les plus caractéristiques d'une forme, comme les zones de changement de courbures qui ont l'avantage d'être assez stables par toute transformation affine.

Il est important de noter que toutes ces primitives doivent être capables de s'adapter aux changements d'échelle.

1.2.3 Organisation des primitives de formes

L'ensemble des primitives peut s'organiser de deux façons:

- dans une hiérarchie où chaque niveau correspond à un degré de précision dans la description (voir [BRO81]). Cette méthode permet par exemple de reconnaître des parties de l'objet avant de le reconnaître dans son entier.
- dans un graphe d'adjacence.

Les deux méthodes ne sont pas incompatibles. On construit ainsi un graphe dans lequel chaque arc va recevoir une information sur la relation entre les deux primitives reliées.

1.2.4 Représenter les modèles

Un modèle de l'objet peut être paramétrisé (par exemple "un angle pour décrire l'ouverture d'une paire de ciseaux", voir [BRO81]), et contenir des contraintes (par exemple "la longueur du cylindre ne doit pas dépasser deux fois la largeur" …, voir [BRO83]). Modéliser ces contraintes en termes de probabilités ajoute une souplesse à l'ensemble.

Certaines particularités de l'objet sont moins longues à détecter que d'autres: on aura donc intérêt à attribuer des coûts en temps à chaque caractéristique des modèles.

1.3 Reconnaissance

La reconnaissance établit des relations entre les modèles d'objets et l'image. En entrée se trouvent la librairie des modèles et l'image, et en sortie l'identité et la position des objets reconnus. **Plusieurs facteurs entravent cette recherche:**

- quels sont les objets présents parmi ceux de la librairie?

– dans quelles positions sont-ils?

– un objet est-il caché partiellement par un autre?

– l'image est elle encombrée d'objets intrus, ou de faux détails ajoutés par la prise de vue?

– la caméra est-elle de bonne qualité?

La méthode pour détecter un seul objet dans une image se décompose en plusieurs étapes:

– *détection de caractéristiques:* on extrait des informations de l'image (par détection de contours ou segmentation d'images généralement).

– *organisation perceptive:* on regroupe ces informations de façon cohérentes (trouver des segments parallèles par exemple).

– *indexation:* on sélectionne l'objet correspondant le mieux à ce groupe d'informations, dans une base de données judicieusement construite.

– *mises en correspondance:* on établit des relations entre chaque caractéristique de l'image et le modèle de l'objet. Cette recherche s'effectue soit dans l'espace des correspondances (arbre des relations images-modèle), soit dans l'espace des transformations (les positions de l'objet ou de la caméra), soit les deux.

– *vérification:* on décide si oui ou non ces correspondances sont suffisantes pour affirmer la présence de l'objet dans l'image.

On répète cela pour tous les objets à reconnaître, si l'on cherche à tous les identifier.

1.4 Conclusion

De nombreux progrès restent à faire dans ce domaine:

– Étendre l'ensemble des objets reconnaissables et rendre capable l'ordinateur d'apprendre de nouvelles formes.

– Tester les systèmes déjà existant sur un grand nombre d'exemples.

– Modéliser le bruit.

– Améliorer la phase de reconnaissance en la rendant plus souple et plus rapide.

Je vais maintenant décrire la voie que j'ai suivie pendant mon stage.

2 Le stage

Si deux caméras permettent en théorie de déduire le relief de deux images, une seule ne le permet pas. D'où l'idée d'introduire une information supplémentaire : le modèle géometrique et cinématique de l'objet que l'on cherche à détecter dans l'image. En effet, quand nous fermons un oeil, nous sommes encore capables de nous déplacer et de saisir des objets, car nous connaissons leurs formes et la distance qui nous sépare d'eux pour les avoir cotoyés auparavant. Le système de perception visuelle doit donc s'aider de modèles 3D et 2D dans sa tâche.

2.1 La méthodologie choisie

Pour le cas particulier de la localisation d'une main dans une séquence d'images, les choix suivants ont été faits:

– Types d'objet à reconnaître: un seul objet, mais articulé: la main.

– Types de données que l'on possède sur l'objet à reconnaître: sa forme (modèle 3D, vue centrée-objet).

– Types d'images: images monoculaires prises par une caméra vidéo qu'on supposera calibrée.

Comme l'une des applications de ce système est la "souris 3D", une interface homme-machine destinée au plus grand nombre de configurations matérielles possibles, il est nécessaire d'utiliser peu de place mémoire. C'est pourquoi le modèle centré-vue a été ecarté.

Le modèle 3D de la main, qui s'inspire de celui de Jim Rehg ([REH95]), est décrit en annexe (voir fig 28 et fig 29).

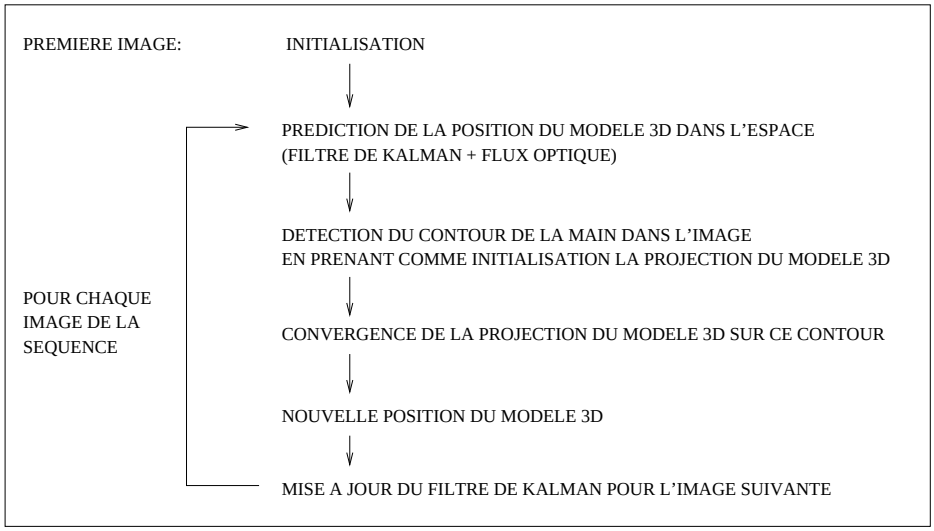


Figure 2 – *Suivi de la main en monoculaire*

La méthode de suivi de la main est la suivante (voir fig 2):

1. Pour la première image, on suppose la position de la main connue grâce à une initialisation préalable
2. Pour les images suivantes, on prédit (grâce à un filtre de Kalman ou du flux optique) la position dans l'espace de la main
3. On projette notre modèle 3D (superposé à cette estimation) sur le plan rétinien (image)
4. On construit le contour de cette projection (voir fig 3)
5. On compare ce contour avec le contour de la main trouvée dans l'image (voir fig 4)
6. L'erreur commise sert à corriger notre filtre de Kalman pour prédire la position de la main dans l'image suivante

C'est un problème très vaste, et nous nous limiterons, dans le cadre de ce stage, à l'étude de l'initialisation dans la première image (étape 1 ci-dessus).

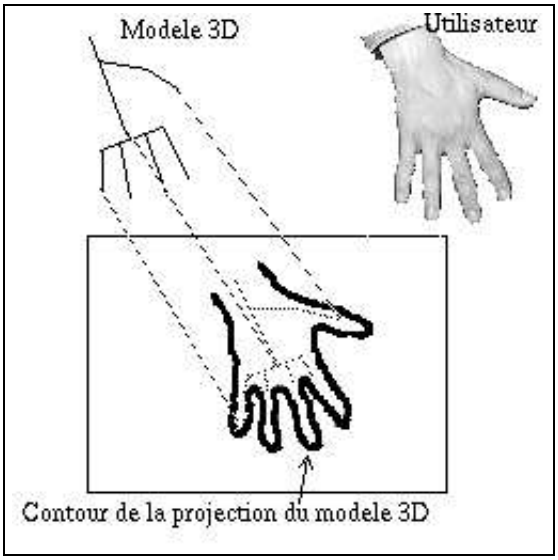


Figure 3 – *Première étape: calculer le contour de la projection du modèle*

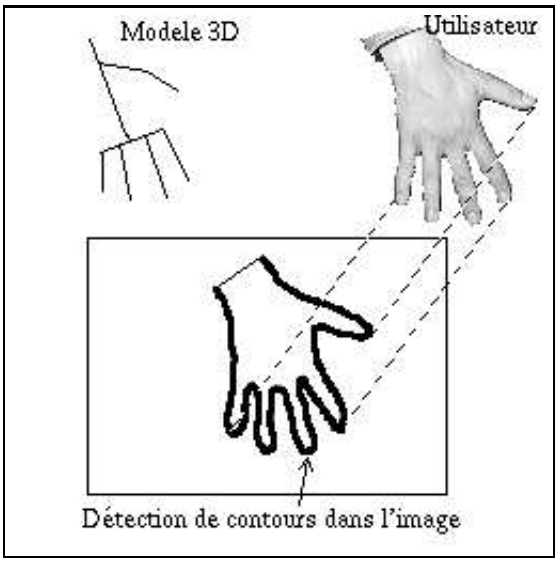


Figure 4 – *Deuxième étape: détecter le contour de la main dans l'image*

2.2 Développements

Le problème de l'initialisation est le suivant: sachant qu'une main possède 27 degrés de liberté, et donc que le nombre de configurations possibles est astronomique, comment détecter une main dans une image? La méthode employée ici consiste à supposer que la main de l'utilisateur est plane dans les premières images (aucun doigt plié) et presque face à la caméra (plan fronto-parallèle). On supposera de plus que la main se trouve dans une certaine région de l'image: par exemple dans un rectangle qui couvre le huitième de l'image. On initialise alors un contour actif sur ce rectangle, pour le faire converger vers la main (voir chapitre suivant). Pour plus de renseignements sur les contours actifs géodésiques, voir en annexe.

2.2.1 Construire un modèle de main

Pour repondre aux besoins du stage, le modèle de la main a besoin de répondre à quelques impératifs:

- Etre composé de volumes simples (cônes,sphères,cubes).
- Etre articulable.
- Pouvoir facilement calculer sa projection sur l'écran.
- Que le contour de cette projection soit proche de la réalité.

Le dernier point est très important: il faut pouvoir comparer le contour de la projection du modèle 3D avec le contour de la main trouvé dans l'image. C'est aussi l'un des éléments les plus difficile à réaliser. En effet, si les doigts peuvent être bien modélisés par des cônes tronqués, la paume pose un problème. La surface de celle-ci est complexe, et sa déformation est difficilement modélisable. C'est pourquoi je me suis limité aux doigts dans le cadre de ce stage.

Le modèle choisi est un modèle hiérarchique composé de cônes tronqués pour les doigts, et de sphères pour les articulations. Les rayons des disques aux extrémités des cônes et des sphères sont égaux sur une même articulation, ce qui assure que la projection du contour des doigts est continue.

Je me suis inspiré de la thèse de Jim Rehg pour évaluer les longueurs des doigts et autres positions d'articulations (voir [REH95] et l'annexe). J'ai aussi adopté son modèle cinématique qui s'inspire lui-même de la robotique.

A partir de la position du repère du centre de la paume, on definit cinq series de changements de repères pour caracteriser l'emplacement des cinq doigts. Chaque transformation s'ecrit sous la forme d'une rotation autour de l'axe des x, d'une translation suivant l'axe des x, puis des z, et enfin d'une rotation autour de l'axe des z: $T_i^{i+1} = R_z(\theta_i).T_z(d_i).T_x(a_i).R_x(\alpha_i)$.

Ces quatres opérations suffisent pour simuler des systèmes articulés.

Pour l'implémentation de ce système, le C++ à été choisi pour sa souplesse. Un module Model.h regroupe toutes les définitions des transformations et des objets. Avec l'aide de Imed Zoghlami (thésard) qui doit modéliser un être humain, j'ai programmé ce module de la facon suivante (voir fig 5):

class Transformation: cet objet contient la description du changement de repère (2 paramètres de rotation et 2 paramètres de translation), ainsi que deux pointeurs: un sur un éventuel objet (sphère, cube, …) attaché à ce repère, et un deuxième pointant sur un tableau de pointeurs sur les changements de repères suivants. Des fonctions membres permettent de manipuler cet objet: création, modification, destruction. D'autres permettent un traitement spécial: Exec_Rec_FORWARD(), Exec_Rec _BACKWARD(), et Exec(). Ces dernières seront décrites ci après.

class Object: cet objet contient un pointeur sur la transformation à laquelle il est attaché. Il contient aussi des fonctions membres pour sa création, modification et destruction. Enfin il contient une fonction Exec() qui est expliquée ci dessous. Cet objet est abstrait, c'est à dire qu'aucune variable ne peut être de ce type.

L'utilisation de ce module se fait de la façon suivante:

On définit une classe dérivée de transformation (ChgtRepere par exemple), et autant de classes dérivées de Object qui serviront a décrire les objets attachés aux repères (sphères, cônes, boites, …). L'avantage d'utiliser le C++ ici est qu'un programmeur peut redéfinir ses propres objets derivés de la classe Object (voir fig 5).

Pour ma part, j'ai definis deux objets : Sphere et Cone. L'affichage de la main à l'écran utilise les fonctions membres virtuelles Transformation::Exec_Rec_ FORWARD(), Transformation::Exec() et Object::Exec() redéfinies dans leurs classes derivées respectives:

– ChgtRepere::Exec() contient les instructions OpenGL de changement de repère (glRotate et glTranslate).

– Sphere et Cone::Exec() contiennent l'affichage OpenGL de ces objets.

– ChgtRepere::Exec_Rec_FORWARD() s'occupe de faire appel à la fonction Exec() de l'éventuel objet attaché à cette transformation, puis de faire appel à sa propre fonction Exec() et enfin de faire appel à toutes les fonctions Exec _Rec_FORWARD() de ses transformations filles.

Ainsi, un simple appel à Base->Exec_Rec_FORWARD(), Base étant un pointeur sur le premier changement de repère de la main, affiche toute la main (voir fig 5).

Il faut maintenant calculer la transformation entre le modèle 3D et la main de l'utilisateur.

2.2.2 Les contours actifs

Pour isoler la main de l'utilisateur dans l'image, nous utilisons les contours actifs géodésiques (voir description en annexe). Ceux-ci convergeront vers le contour de la main. Toutefois un problème survient: le contour actif a tendance à rentrer dans la main au niveau du poignet. Il faut donc faire une hypothèse supplémentaire: l'un des côté du rectangle est fixe, il n'évolue pas, et l'on suppose que ce côté correspond au poignet. Pour fixer cette partie du contour, on modifie l'image en ajoutant un pic d'intensité tout du long (voir figures dans le chapitre 2.3).

Les géodésiques ont été choisis parce qu'ils effectuent deux étapes simultanément dans la phase d'extraction d'informations dans l'image: la phase de détection de contours et la phase d'organisation de ces primitives en une courbe fermée régulière. Toutefois ils ont quelques défauts:

– avoir le contour de la main de l'utilisateur ne suffira probablement pas pour localiser avec précision la position dans l'espace de la main.

– un faible changement dans l'image peut modifier sensiblement le contour.

– l'arrière plan doit être le plus uniforme possible pour que le géodésique converge efficacement.

Malgré cela, beaucoup de renseignements utiles sont fournis par ce contour actifs:

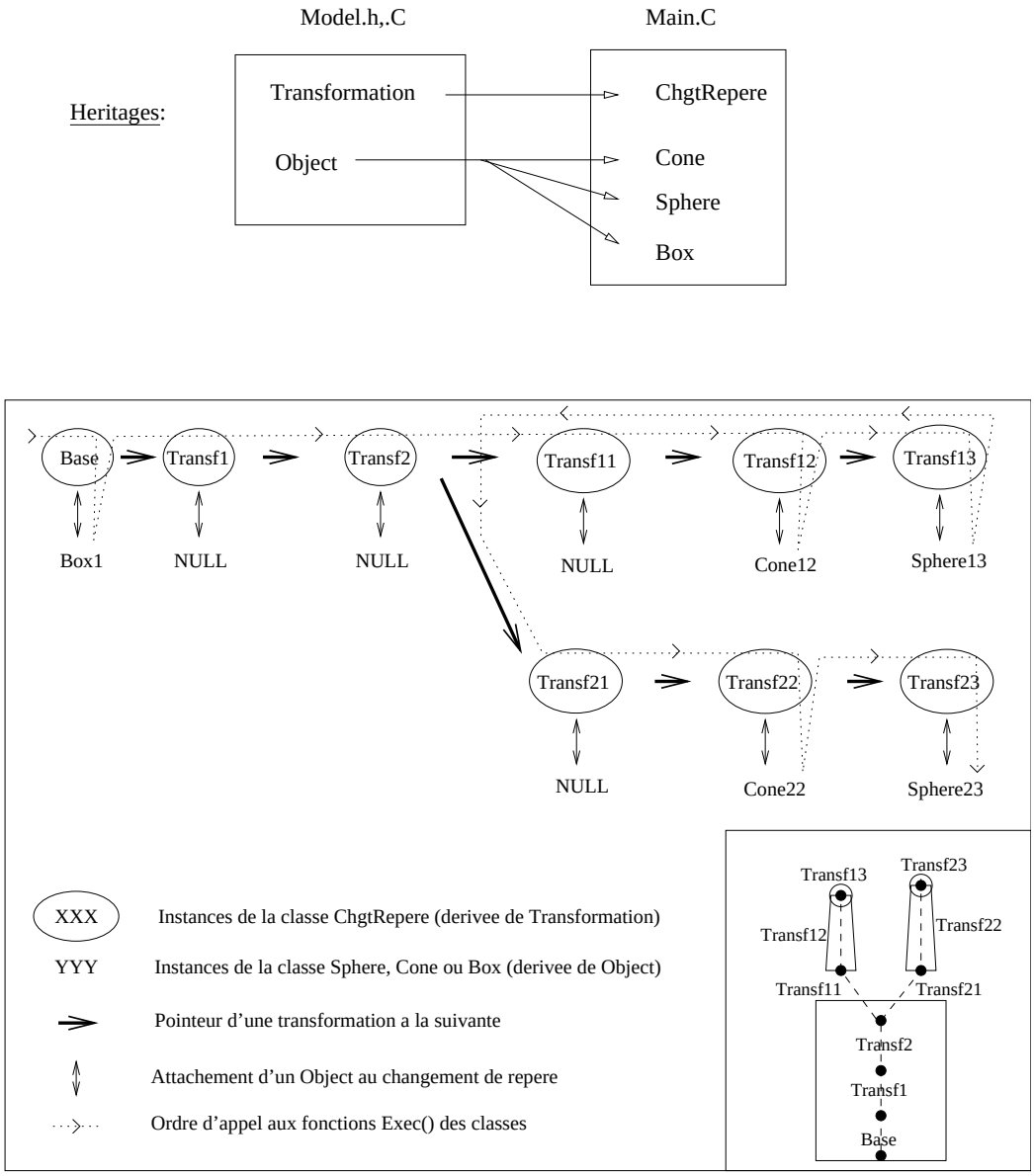


Figure 5 – *Un exemple de structure utilisant Model.h*

2.2.3 Extraire des points d'intérêt grâce au contour

Pour estimer la rotation R et la translation t qui font passer le modèle 3D de sa position initiale à la position estimée de la main dans l'image, on met en correspondance quatre points significatifs:

– Première paire de points: l'extrémité du pouce.

– Deuxième paire de points: l'extrémité de l'index.

– Troisième paire de points: l'extrémité de l'auriculaire.

– Quatrième paire de points: la base du pouce sur le poignet.

Ces quatres points ont été choisis parce qu'ils ont peu de chance d'être alignés, ce qui serait une source d'erreur pour estimer R et t. Pour le modèle 3D, calculer la projection de ces 4 points sur le plan rétinien est trivial (on suppose que la caméra est calibrée). Pour l'image, on se sert du contour géodésique trouvé:
- On lisse ce contour (x(t),y(t)) par un filtre gaussien.
- On trace la courbe de l'angle que fait la tangente avec un axe horizontal.
- On deplace une fenêtre le long de cette courbe pour détecter les maxima de courbure (voir fig 6).

La fenêtre fonctionne comme suit: pour chaque point de la courbe, on fait la moyenne des angles des points situés entre -Tmax et -Tmin, et la moyenne entre Tmin et Tmax par rapport au point courant. Si ces deux moyennes sont de part et d'autre de l'intervalle [-Seuil,Seuil] centré autour du point courant, on marque ce point comme un point de forte courbure. Puis on fait la moyenne des angles de chacune des zones de forte courbure trouvées: ce sont les maxima de courbure. On doit en trouver 11 (2 pour le poignet, 5 pour chaque doigt et 4 entre les doigts).

On trie ces maxima pour en extraire ceux cherchés. Pour cela, on remarque que quatre maxima de courbure qui se suivent et qui sont de même signe correspondent forcément au passage du poignet. On remarque ensuite que dans un sens de parcours sur le contour, deux maxima successifs et de sens opposés sont plus éloignés que les autres: ils correspondent au passage du pouce. Il est facile ensuite d'en déduire où se trouvent chacun des doigts.

On possède donc la projection des 4 points d'intérêt de la main de l'utilisateur.

Le problème est que le modèle 3D n'est jamais la réplique exacte de la main de l'utilisateur. De plus les données sont bruitées.

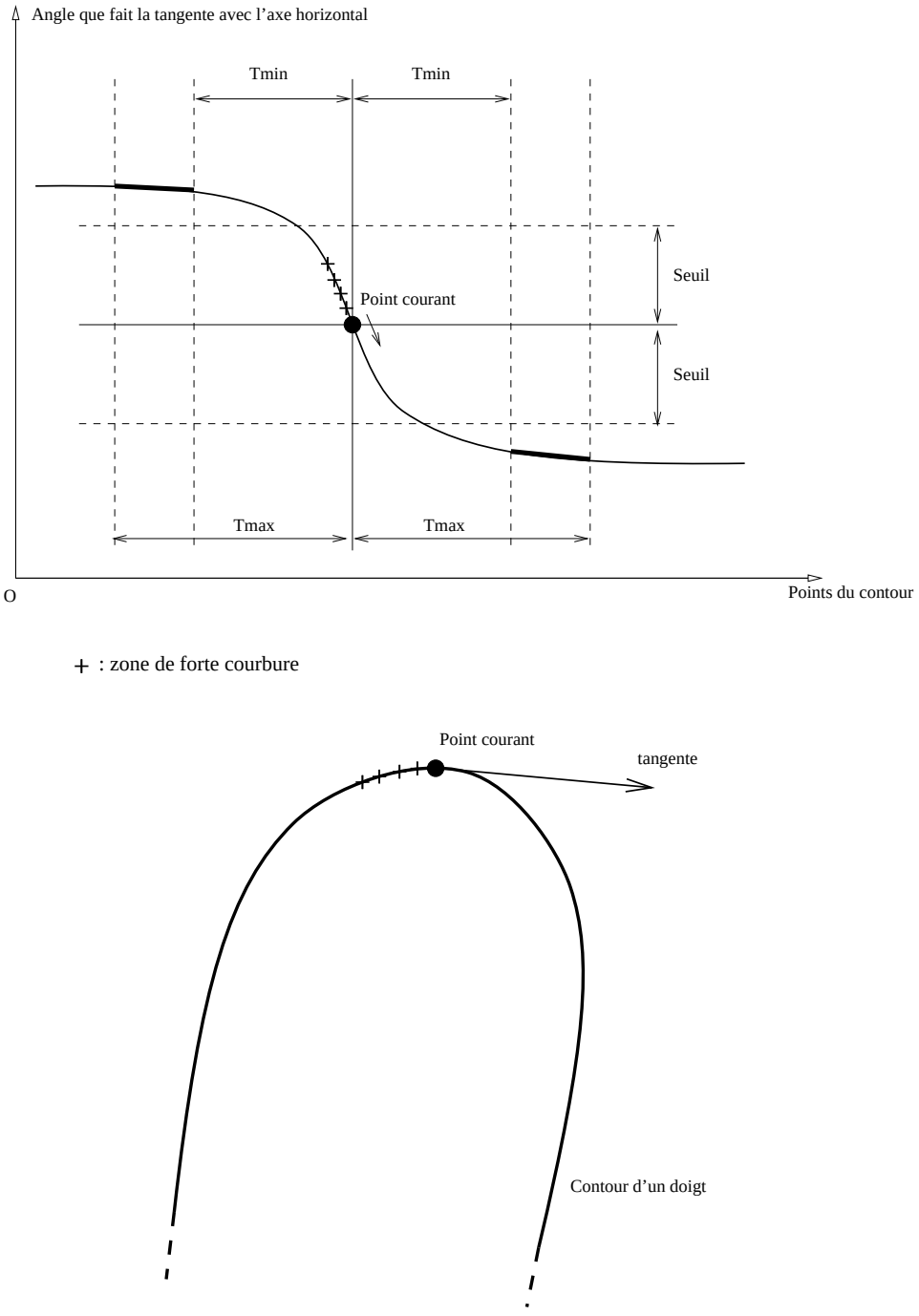


Figure 6 – *Calculer les zones de plus forte courbure le long du contour*

Comment estimer R et t à partir de ces quatre correspondances?

2.2.4 Estimer la transformation R,t

Soient (voir fig 7):

- m_i un point du modèle 3D projeté $(x_i, y_i, 1)$.
- m'_i un point de la main réelle projetée $(x'_i, y'_i, 1)$.
- R la rotation cherchée (θ, ϕ, ψ).
- t la translation cherchée (t_x, t_y, t_z).
- M_i le point du modèle 3D (X_i, Y_i, Z_i).
- M'_i le point de la main réelle (X'_i, Y'_i, Z'_i) (inconnu).

On a les relations suivantes:

$$M'_i = R.M_i + t$$

$$m_i = 1/Z_i.M_i$$

$$m'_i = 1/Z'_i.M'_i$$

On cherche R et t qui minimisent:
$\sum_{i=1}^{i=4} \parallel M'_i - (R.M_i + t) \parallel^2$
c'est à dire:
$\sum_{i=1}^{i=4} \parallel Z'_i.m'_i - (R.M_i + t) \parallel^2$

On écrit R sous la forme du produit de trois matrices de rotations par rapport à l'axe des x, l'axe des y et l'axe des z:
$R(\theta, \phi, \psi) = R_z(\theta).R_y(\phi).R_x(\psi)$

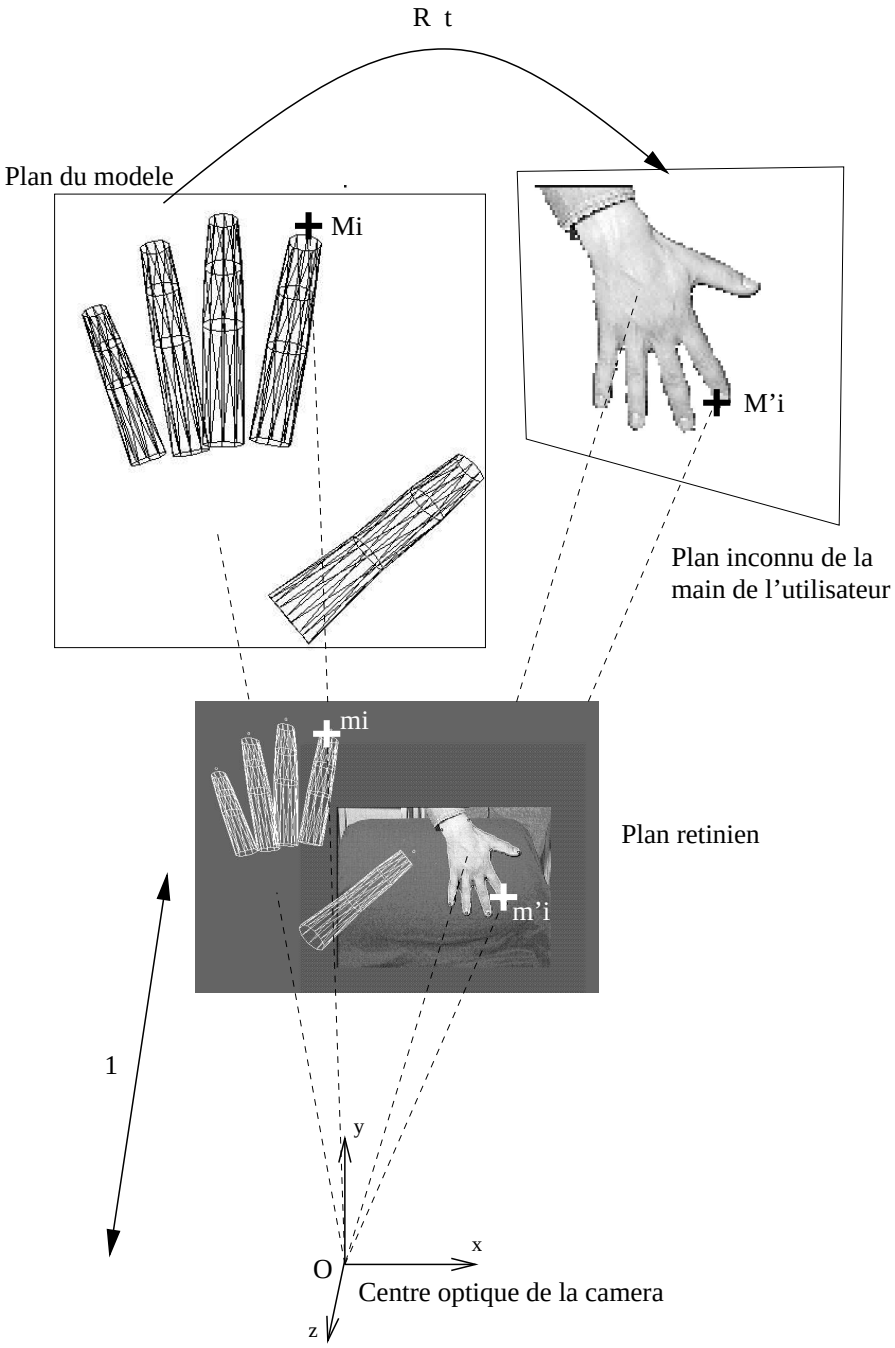


Figure 7 – *Estimer la transformation rigide R,t entre le modèle et la main*

avec:

$$R_x(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\psi) & -sin(\psi) \\ 0 & sin(\psi) & cos(\psi) \end{pmatrix}$$

$$R_y(\phi) = \begin{pmatrix} cos(\phi) & 0 & sin(\phi) \\ 0 & 1 & 0 \\ -sin(\phi) & 0 & cos(\phi) \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

d'où (c représente cos et s sin par manque de place):

$$R(\theta,\phi,\psi) = \begin{pmatrix} c(\phi).c(\theta) & -c(\psi).s(\theta)+s(\psi).s(\phi).c(\theta) & s(\psi).s(\theta)+c(\psi).s(\phi).c(\theta) \\ c(\phi).s(\theta) & c(\psi).c(\theta)+s(\psi).s(\phi).s(\theta) & -s(\psi).c(\theta)+c(\psi).s(\phi).s(\theta) \\ -s(\phi) & s(\psi).c(\phi) & c(\psi).c(\phi) \end{pmatrix}$$

et donc le critère C à minimiser au sens des moindres carrés s'écrit:

$$\begin{aligned} & C(\theta,\phi,\psi,t_x,t_y,t_z,Z'_1,Z'_2,Z'_3,Z'_4) = \\ & \textstyle\sum_{i=1}^{i=4}(\\ & (Z'_i.x'_i \quad - \quad ((\ cos(\phi).cos(\theta).X_i \\ & \qquad\qquad + \quad (-cos(\psi).sin(\theta)+sin(\psi).sin(\phi).cos(\theta)).Y_i \\ & \qquad\qquad + \quad (sin(\psi).sin(\theta)+cos(\psi).sin(\phi).cos(\theta)).Z_i) \\ & \qquad\qquad + \quad t_x))^2 \\ & +(Z'_i.y'_i \quad - \quad ((\ cos(\phi).sin(\theta).X_i \\ & \qquad\qquad + \quad (cos(\psi).cos(\theta)+sin(\psi).sin(\phi).sin(\theta)).Y_i \\ & \qquad\qquad + \quad (-sin(\psi).cos(\theta)+cos(\psi).sin(\phi).sin(\theta)).Z_i) \\ & \qquad\qquad + \quad t_y))^2 \\ & +(Z'_i \quad - \quad ((\ -sin(\phi).X_i \\ & \qquad\qquad + \quad sin(\psi).cos(\phi).Y_i \\ & \qquad\qquad + \quad cos(\psi).cos(\phi).Z_i) \\ & \qquad\qquad + \quad t_z))^2 \\ &) \end{aligned}$$

En fait, il est possible de réduire le nombre de paramètres de minimisation en remplaçant tous les Z'_i par un seul et même paramètre. En effet, si l'on a supposé

que la main était presque dans un plan fronto-parallèle, on peut admettre que tous les Z'_i sont égaux. On peut même aller plus loin en disant que cette moyenne des Z'_i est proche de la translation cherchée suivant l'axe des z. Cela revient approximativement à travailler sur le barycentre des points, et les résultats prouvent que cela suffit pour une première approximation de la position de la main (voir résultats dans le chapitre 2.3).

L'étape suivante consiste à superposer au mieux les doigts du modèle 3D avec les doigts de l'image. Pour cela, on va minimiser la distance entre les points extrémités correspondant, en fonction de l'angle à la base de chaque doigt. Ici aussi, nous supposerons que les doigts de l'utilisateur restent dans le plan de la paume.

2.3 Les résultats

2.3.1 Le modèle 3D de la main

Pour avoir une idée de l'efficacité de notre modèle 3D, j'ai utilisé les librairies graphiques 3D OpenGL. Ces fonctions C ont l'avantage d'être simples à utiliser, adaptées à l'affichage de modèles hiérarchiques, et bien documentées.

Les phallanges sont constituées de cônes tronqués, et les articulations sont des sphères (voir figures 8 et 9).

2.3.2 La convergence des contours actifs géodésiques

Cette convergence s'effectue très bien quand l'arrière plan est uniforme (voir figures 10 et 11).

L'algorithme construit régulièrement, pendant la convergence, une carte des distances, qui représente les distances de points du plan au snake (voir lignes de niveaux figures 12). Ce sont ces cartes qui pourront être utilisées pour améliorer la convergence du modèle 3D sur l'image.

Par contre, si le contraste est trop faible, ou si l'arrière plan est trop hétérogène, les problèmes arrivent rapidement (voir figures 13)!

Il a fallu modifier l'algorithme d'évolution des géodésiques pour fixer une partie du contour (voir figure 14).

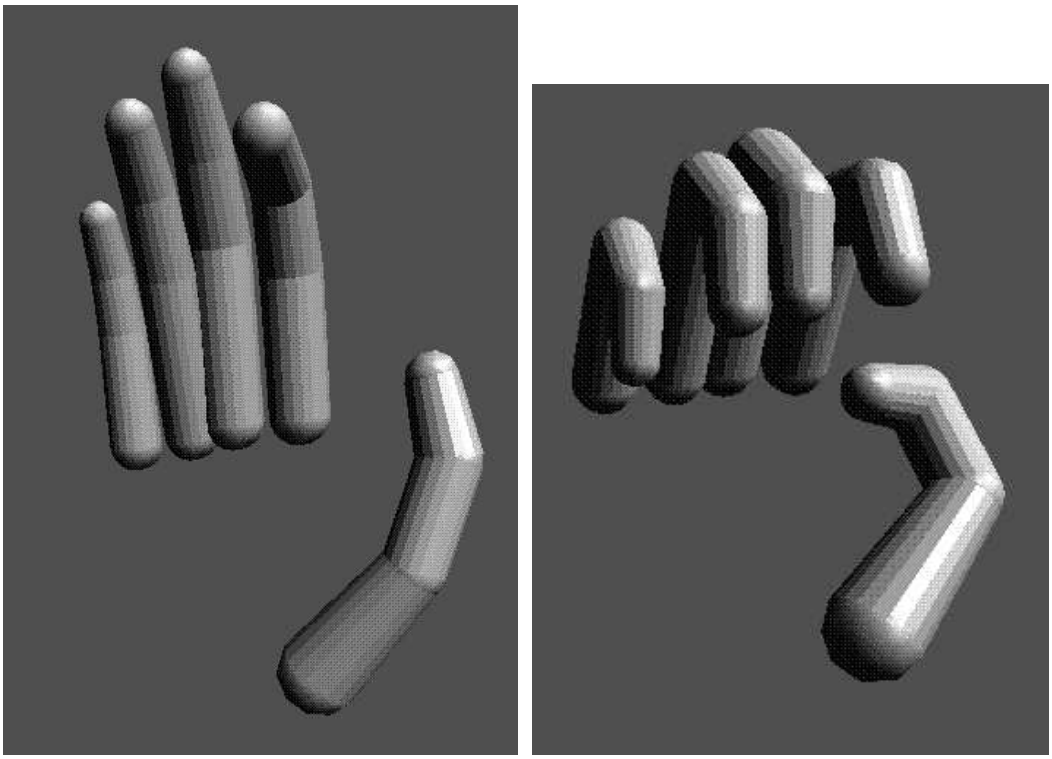


Figure 8 – *Deux positions du modèle 3D de la main (OpenGL)*

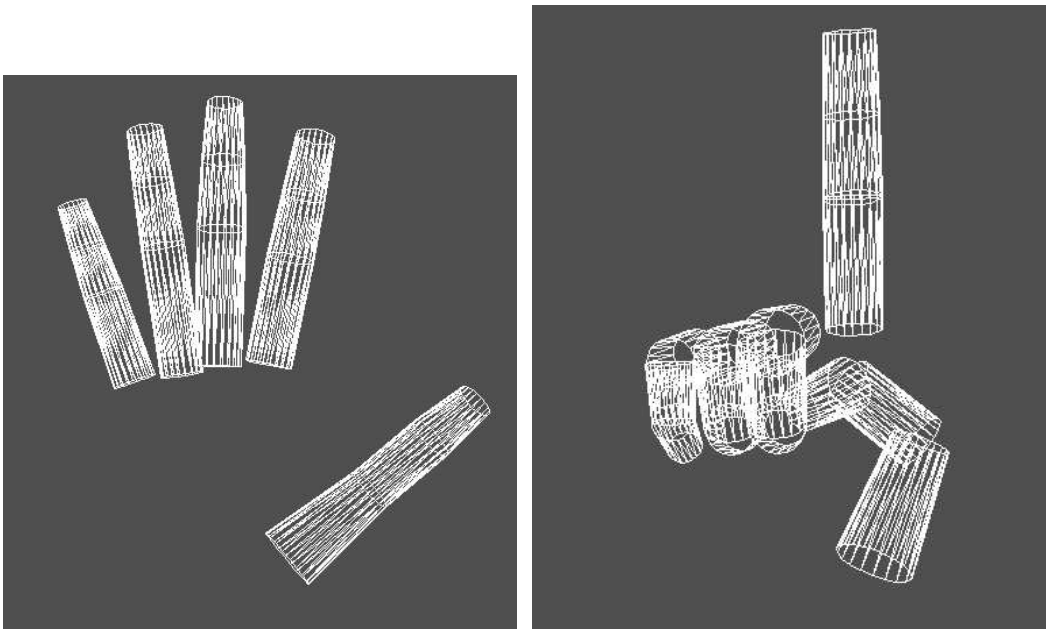


Figure 9 – *Deux positions du modèle 3D fil-de-fer de la main (OpenGL)*

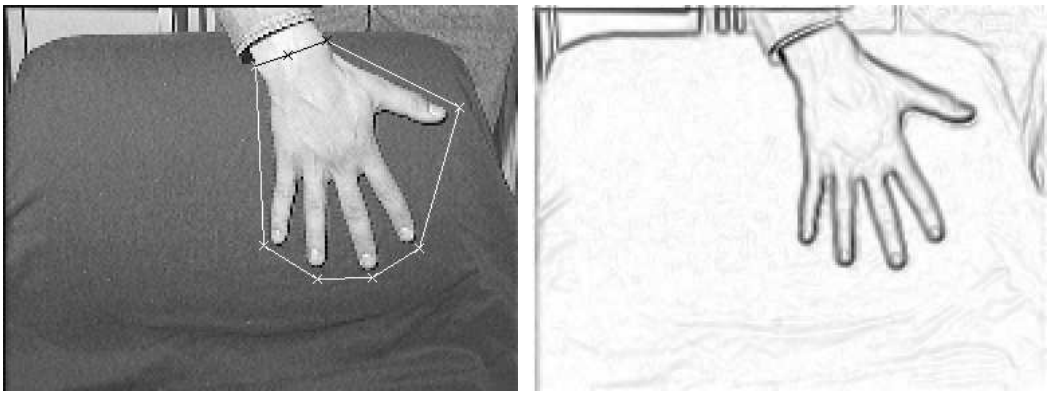


Figure 10 – *Figure de gauche: Image et initialisation du contour géodésique (partie fixe en noir), figure de droite: gradient de l'image*

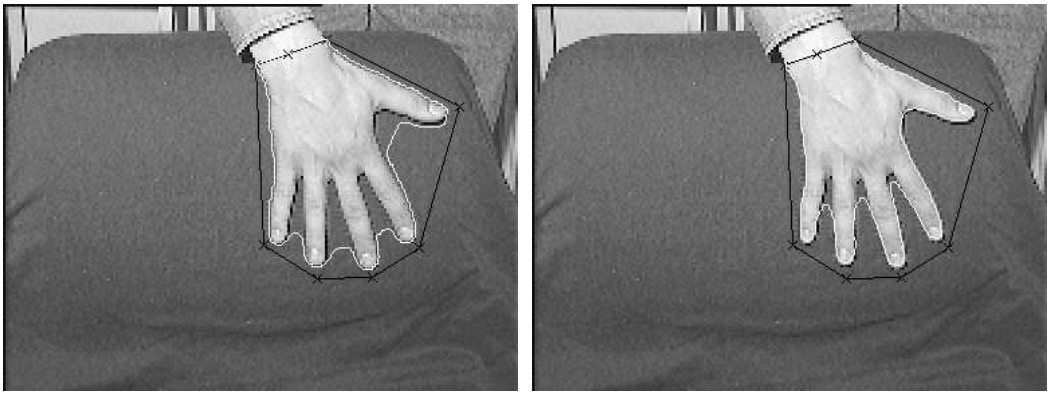


Figure 11 – *Convergence du snake: figure de gauche au bout de 200 itérations, figure de droite au bout de 600 itérations*

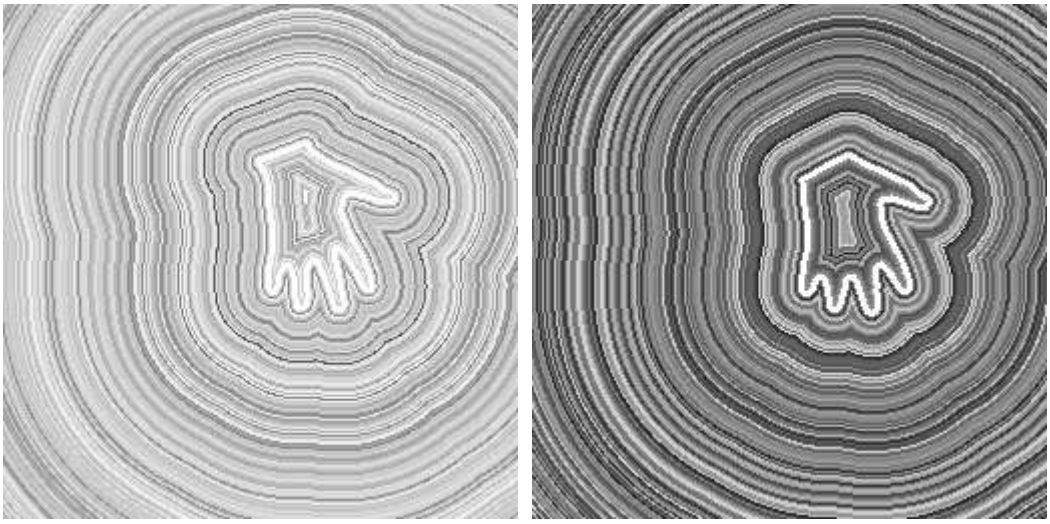


Figure 12 – *La carte des distances*

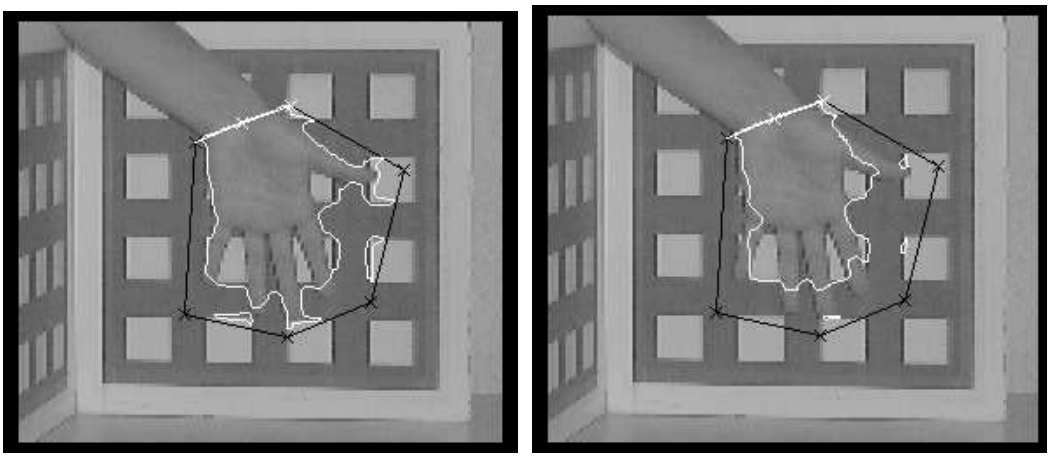


Figure 13 – *Figure de droite: problèmes de convergence après 200 itérations (initialisation en noir), figure de gauche: après 400 itérations.*

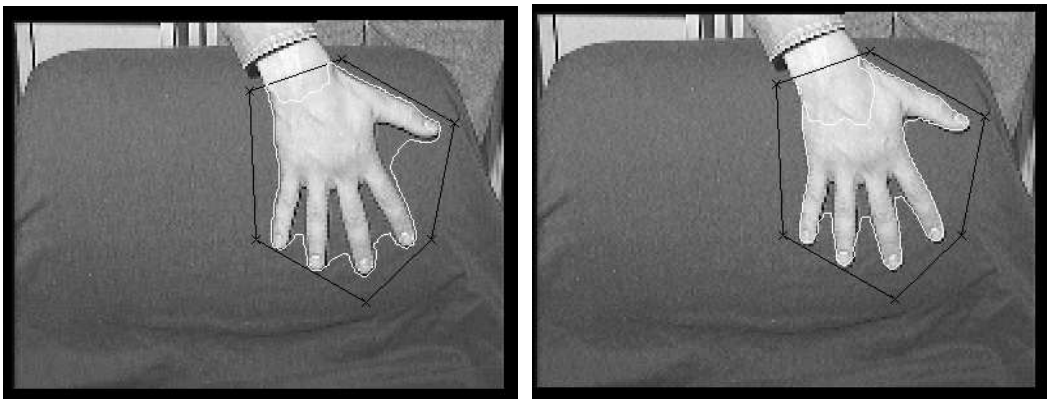


Figure 14 – *Le problème du poignet (200 et 500 iterations)*

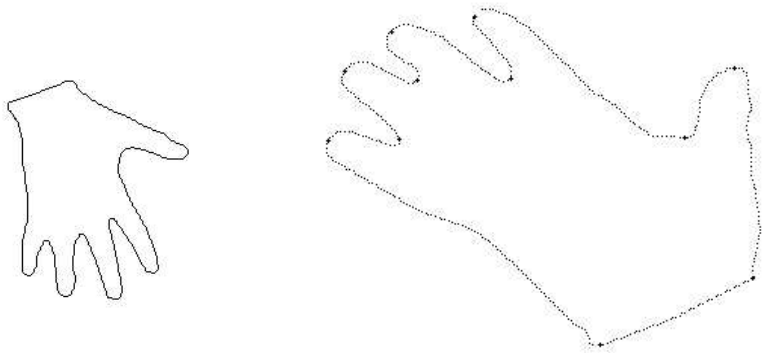


Figure 15 – *Deux snakes extraits de deux images*

2.3.3 L'extraction des points d'intérêt

Une fois que l'on possède la liste des points du contour de la main, il est possible de repérer des points particuliers de la main de l'utilisateur. L'algorithme de détection des maxima de courbure fonctionne bien (voir figures 15, 16 et 17).

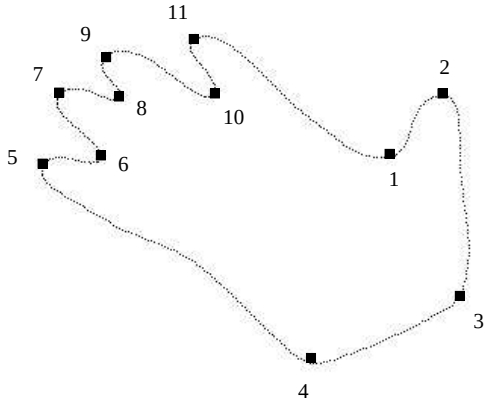


Figure 16 – *Un contour lissé et les maxima de courbure trouvés*

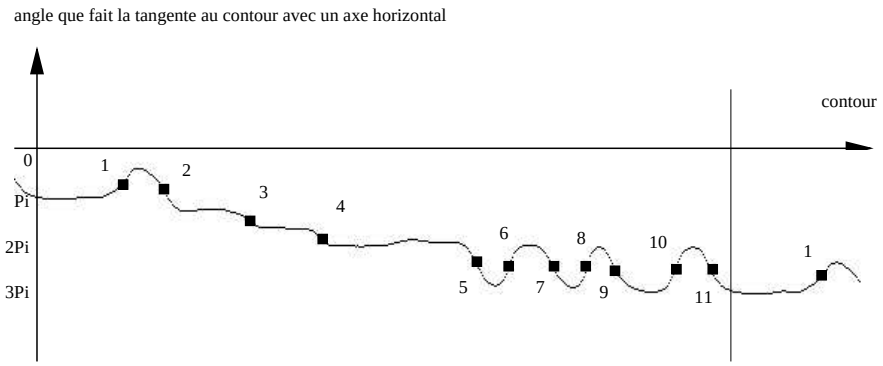


Figure 17 – *La courbe de l'angle que fait la tangente au contour avec un axe horizontal et les maxima de courbure trouvés*

2.3.4 L'estimation de R et t

L'estimation de R et t est assez bonne (voir figure 18, 20 et 22). Mais l'algorithme a du mal à converger quand le modèle doit faire un demi-tour par exemple (voir figure 24)

Voici les données fournies par l'algorithme de minimisation MiniSimplex (qui minimise la valeur d'une fonction):

Les 6 valeurs sont: les translations x,y,z et les rotations θ, ϕ, ψ. La valeur de la fonction est la somme des carrés des distances entre les quatre points caractéristiques du modèle et de l'image.

– Figure 18:

  Initial values: 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00
  Function value: 6.66488e+00

  After 933 evaluations of the objection function ...
  Final values: 2.86519e+00 -3.98706e-03 2.40096e-01 -5.37656e-01 -7.07196e-01 -4.59570e+00
  Function value: 2.56884e-02

– Figure 20:

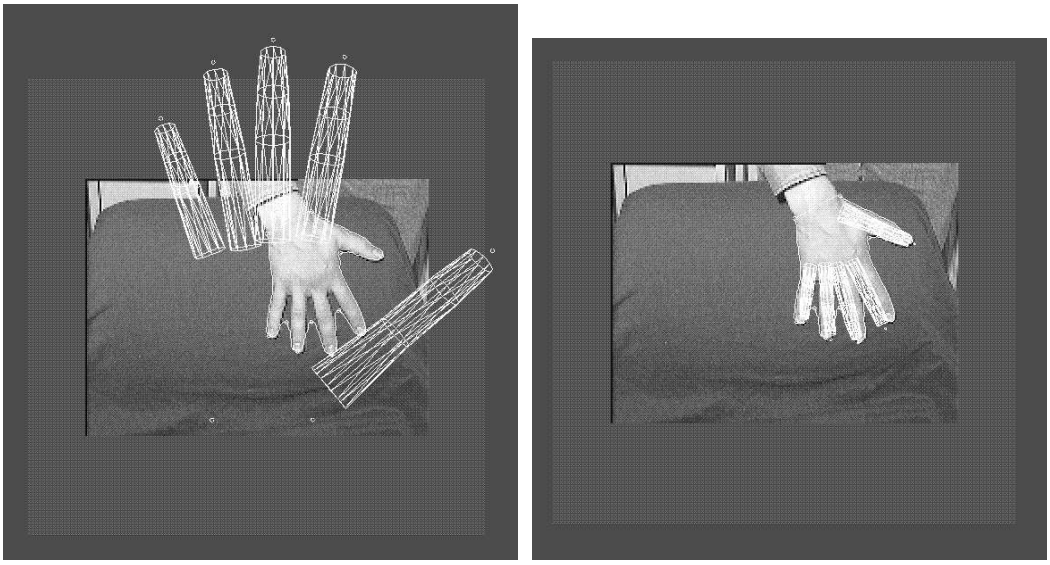


Figure 18 – *Estimation de la posture de la main (initialisation+résultat)*

Figure 19 – *Estimation de la position des doigts*

Initial values: 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+0
0 0.00000e+00
Function value: 3.03791e+00

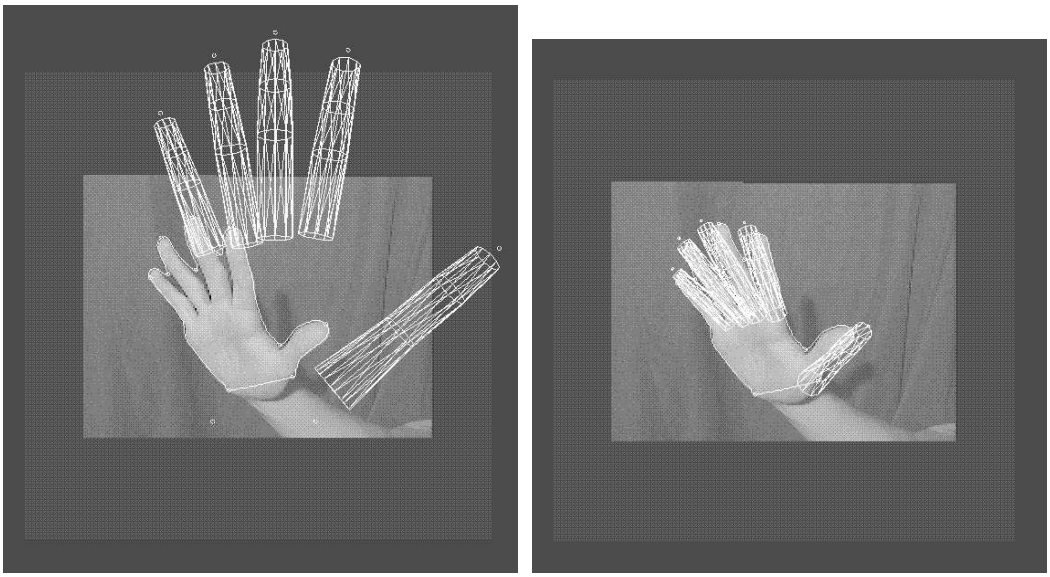


Figure 20 – *Un autre exemple d'estimation de la posture de la main (initialisation+résultat). Ici la position de la paume est mal estimée, car le contour actif coupe la main trop haut, ce qui fausse l'estimation de la position du poignet.*

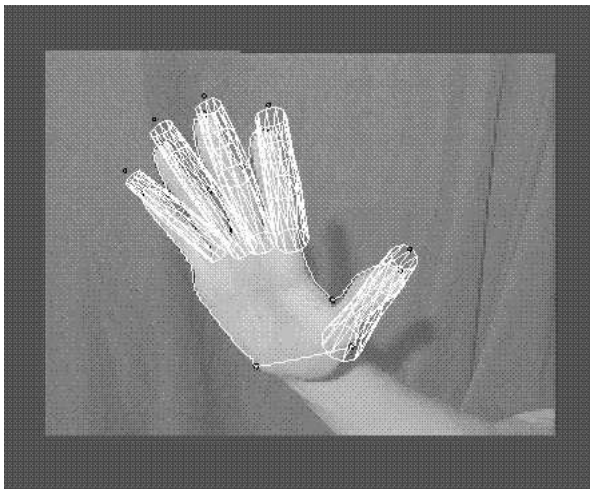


Figure 21 – *Estimation de la position des doigts*

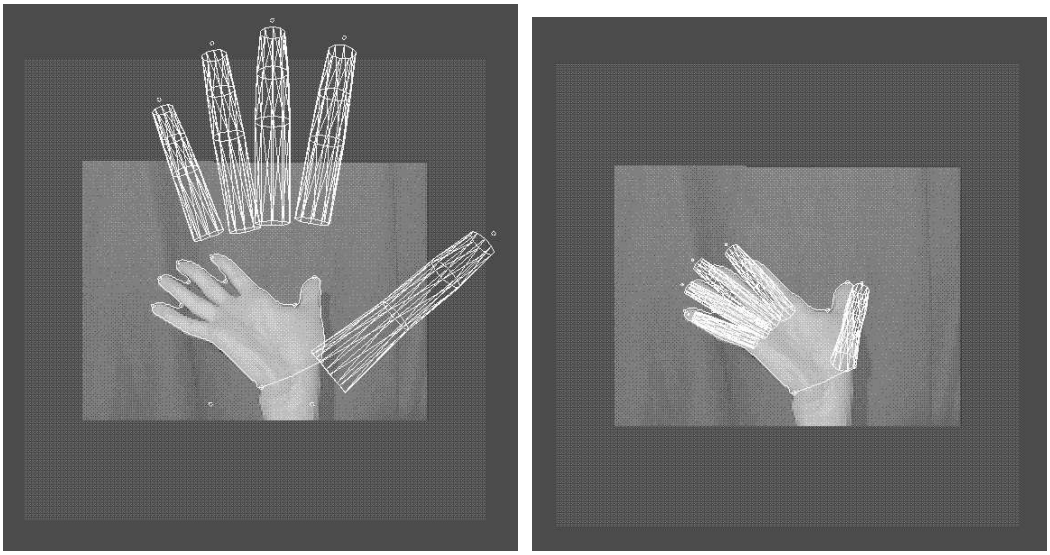


Figure 22 – *Un autre exemple d'estimation de la posture de la main (initialisation+résultat)*

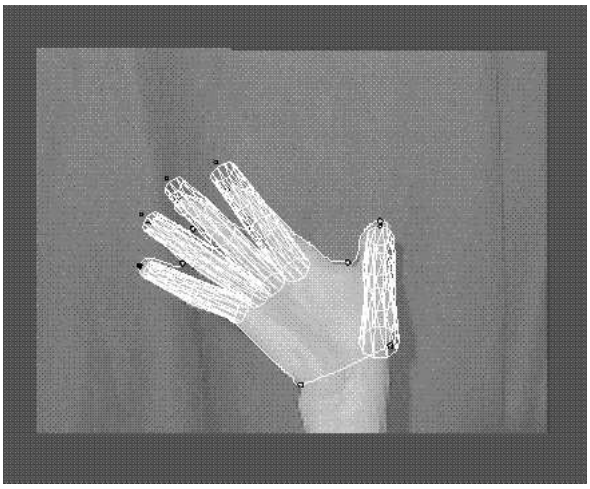


Figure 23 – *Estimation de la position des doigts*

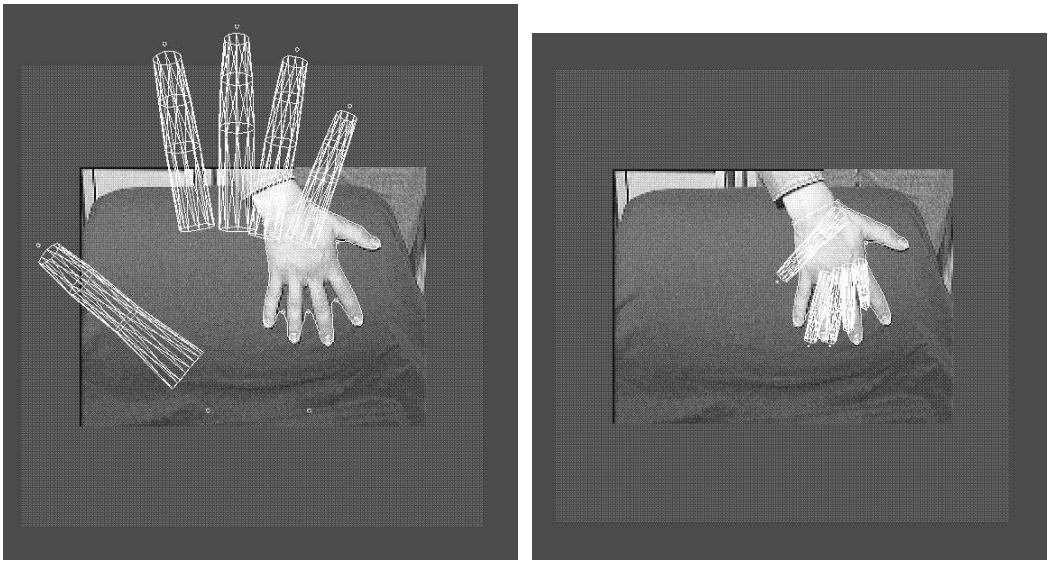


Figure 24 – *Echec dans l'estimation de la posture de la main (initialisation différente+résultat)*

After 571 evaluations of the objection function ...
Final values: 7.09092e-01 -5.44217e-01 8.36080e-02 1.87713e-01 3.29276e-01 -2.46810e+00
Function value: 3.66459e-02

– Figure 22:

Initial values: 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00
Function value: 4.31947e+00

After 857 evaluations of the objection function ...
Final values: 5.07501e-01 -5.42368e-01 6.01342e-01 -3.69750e-02 5.57627e-01 -3.03607e+00
Function value: 1.70957e-02

– Figure 24: un exemple d'echec:

Initial values: 0.00000e+00 3.14159e+00 0.00000e+00 0.00000e+00 0.00000e+0
0 0.00000e+00
Function value: 7.04135e+00

After 1653 evaluations of the objection function ...
Final values: 2.66084e+00 3.76883e+00 1.99261e-01 -8.54806e-01 -6.92971e-
01 -3.94918e+00
Function value: 7.84032e-02

Ici, l'algorithme de minimisation a trouvé un minimum local. On remarque sur les figures que les points correspondants au poignet et au majeur du modèle et de l'image sont bien superposés, contrairement aux autres.

Les raisons sont que l'algorithme de minimisation employé n'est pas le meilleur, et que le modèle 3D est trop différent de la main. En effet, il existe de multiples minima locaux autour du minimum global, et étant donné le nombre de paramètres (6), il est très difficile de surmonter cette difficulté. Il y a deux solutions : trouver un algorithme plus efficace, et initialiser la position du modèle 3D plus proche de la solution grâce à des astuces (comme se servir des points d'intérêt pour connaitre le sens de la main de l'utilisateur).

Quant à l'estimation de la position des doigts, on note qu'elle n'est efficace que si la position de la paume de la main a été elle-même bien estimée. C'est pourquoi utiliser la carte des distances créée par l'évolution des contours actifs géodésiques améliorera sûrement l'algorithme (voir chapitre suivant).

3 Les perspectives

3.1 Améliorer les méthodes suivies pendant le stage

Une fois la position de la main estimée, on peut affiner l'estimation de la position et de la taille des doigts en faisant converger non plus des points d'intérêts (extrémités des doigts et poignet) mais tout le contour de la main. Pour cela, on va s'aider d'une particularité que possèdent les contours géodésiques.

Lors de la convergence du contour actif dans l'image, le programme construit une carte des distances (voir définition en annexe). Cette image 2D associe à chaque point (x,y) la valeur de la distance z qui sépare ce point du contour actif courant. Ainsi, si l'on calcule le contour de la projection du modèle 3D, que l'on fait la somme

sur ce contour des distances au géodésique et que l'on minimise cette somme en jouant sur la position des doigts, et leurs tailles, on aura affiné notre modèle 3D.

Pour calculer le contour de la projection du modèle 3D de la main, on peut tracer les silhouettes de chacun des objets projetés sur l'écran, et trouver leur union. Des algorithmes permettent cela: division-fusion, droite de balayage, calcul d'intersections, etc.

On peut encore améliorer l'algorithme de minimisation pour estimer R et t (transformation entre le modèle 3D et la main réelle).

Le suivit du mouvement de la main est un autre problème:

L'étude du flux optique peut fournir des informations intéressantes sur le déplacement de la main dans l'image.

On peut enfin estimer directement la vitesse de la main à partir du déplacement de sa projection dans l'image, et de sa position estimée précédente. En effet, il est possible d'utiliser le fait qu'un doigt se plie dans un plan.

3.2 Autre voie possible: la stéréovision

Contrairement au cas précédent, deux images prises d'une même scène, au même instant et de deux points de vues légérement espacés, fournissent des informations sur le relief de la scène.

On peut alors directement comparer le modèle 3D (et non plus sa projection) avec la reconstruction 3D de la main de l'utilisateur déduite des images (voir fig 25). Pour faciliter cette comparaison, on pourra même utiliser les surfaces géodésiques 3D.

La reconstruction des mains utilise la méthode suivante:

- la rectification: les deux images sont projetées sur un même plan face à la caméra.
- la corrélation: on recherche les points communs dans les deux images.
- la disparité: grâce à la corrélation, on évalue la profondeur de chaque point de la main.

Les figures 26 et 27 représentent deux paires d'images d'une main, et les deux cartes de diparités correspondantes.

Ces résultats sont intéressants.

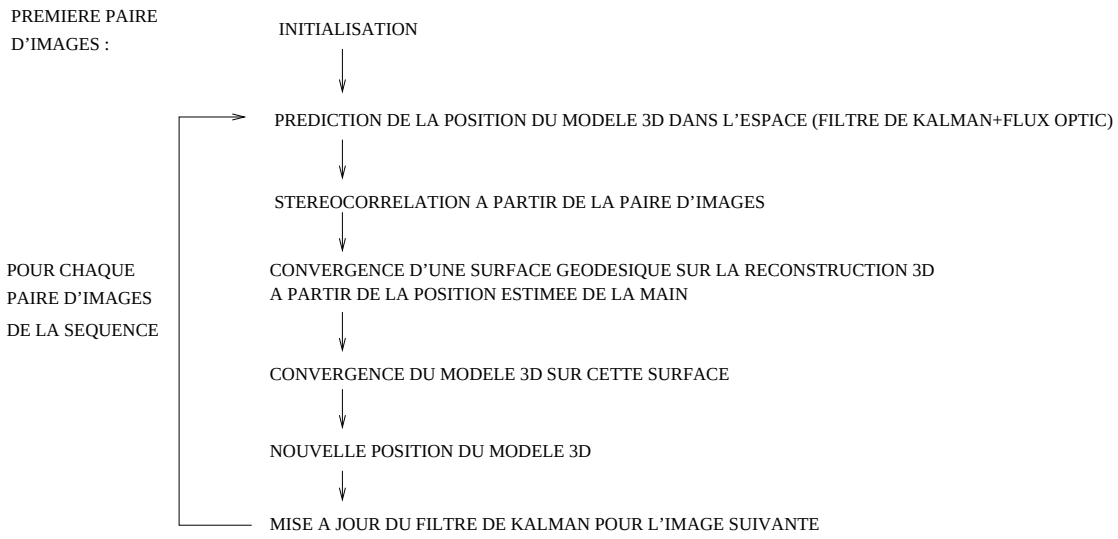


Figure 25 – *Suivi de la main en stéréovision*

Figure 26 – *Deux paires d'images*

Figure 27 – *Les deux cartes de disparités obtenues (un point est d'autant plus foncé qu'il est proche de la caméra)*

Conclusion

Le suivi de la main dans une séquence d'images est un problème très difficile. Pour la partie initialisation, nous avons vu qu'il était possible d'estimer la position dans l'espace de la main de l'utilisateur par rapport à la caméra. Toutefois quelques conditions doivent être remplies: la main doit être plane, et plus ou moins face à la caméra, et l'arrière-plan doit être uniforme. Si les deux premières conditions ne sont pas très contraignantes dans une applications de type "souris 3D" (mettre la main face à la caméra signifie vouloir utiliser ce système par exemple), la dernière pose un problème. Dans ce cas, un pré-traitement de l'image sera nécessaire.

Il est clair que le système final utilisera différentes méthodes complémentaires. Les contours géodésiques fournissent déjà la silhouette de la main. L'étude de l'intérieur de ce contour fournira d'autres informations utiles. La stéréovision est une alternative intéressante.

Beaucoup de travail reste à faire dans ce domaine en pleine explosion.

ANNEXES

Les contours actifs géodésiques:

Cette partie contient une (très) brêve description des contours actifs géodésiques. Pour plus de détails, voir [SET95, MSV93, MSV95], ainsi que [CCD93, CKS94, CKS95, DER95]

Evolution de courbes dans le plan

Il peut être intéressant, pour diverses applications, de faire évoluer des courbes planes.

Soit la courbe $C(p,t)$, nous voulons la faire évoluer dans le temps suivant l'EDP:

$$\left\{ \begin{array}{rcl} \frac{\partial C}{\partial t} & = & F.\vec{N} \\ C(p,0) & = & C_0(p) \end{array} \right. \quad (1)$$

F est une fonction qui dépend par exemple de la courbure de la courbe.

Le problème est que la discrétisation de cette formulation pour le calcul par ordinateur entraine une grande instabilité, et qui plus est interdit tout changement de topologie. C'est pourquoi la courbe C est vue maintenant comme la coupe d'une surface par un plan. On choisit généralement comme surface Φ: $\Phi(x, y)$ = distance du point x,y à la courbe C (positive si à l'extérieur, négative sinon). C'est la carte des distances.

La surface Φ évolue suivant l'EDP:

$$\left\{ \begin{array}{rcc} \Phi_t & = & F.|\nabla\Phi| \\ \Phi_{(x,t=0)} & = & \Phi_{(x,0)} \end{array} \right. \tag{2}$$

La détection de contours dans une image

Dans l'équation 3.1, on décide de faire dépendre F de trois paramètres: la courbure κ, un terme de vitesse ν et le gradient dans l'image $|\nabla I|$. La valeur ν sert à faire évoluer la courbe dans les zones de faible courbure.

L'équation 3.1 devient donc:

$$\left\{ \begin{array}{rcc} \frac{\partial C}{\partial t} & = & g(|\nabla I|).(\nu + \kappa).\vec{N} \\ C(p,0) & = & C_0(p) \end{array} \right. \tag{3}$$

Si l'on note $u(x, y)$ une image de distance dans laquelle la courbe de niveau $u(x, y) = 0$ représente C(p,t), et si l'on remarque que $\kappa = div(\frac{\nabla u}{|\nabla u|})$, on peut montrer l'EDP suivante:

$$\left\{ \begin{array}{rcc} \frac{\partial u}{\partial t} & = & g(|\nabla I|).(\nu + div(\frac{\nabla u}{|\nabla u|})).|\nabla u| \\ u(x,y,0) & = & u_0(x,y) \end{array} \right. \tag{4}$$

Si l'on cherche à construire des contours actifs s'arrêtant sur des forts gradients dans l'image, on peut poser $g(|\nabla I|) = \frac{1}{1+|\nabla I|^2}$.

Une autre approche variationnelle (qui justifie son qualificatif de géodésique) est basée sur la minimisation d'une énergie (voir [CKS94, CKS95]):

$E(C) = \alpha \int_0^1 |C'(p)|^2 dp + \beta \int_0^1 |C''(p)|^2 dp - \lambda \int_0^1 |\nabla I(C(p))|^2 dp$

Elle conduit au schéma suivant, généralisé par le paramètre ν qui accélère la convergence:

$$\begin{cases} \frac{\partial u}{\partial t} & = & g(|\nabla I|).(\nu + div(\frac{\nabla u}{|\nabla u|}).|\nabla u| + \nabla g(|\nabla I|).\nabla u \\ u(x,y,0) & = & u_0(x,y) \end{cases} \quad (5)$$

Voir les exemples de convergence dans la section résultats du chapitre précédent.

Le modèle de la main:

Ce modèle s'inspire de celui de Jim Rehg dans sa thèse ([REH95]). Voir figures pages suivantes.

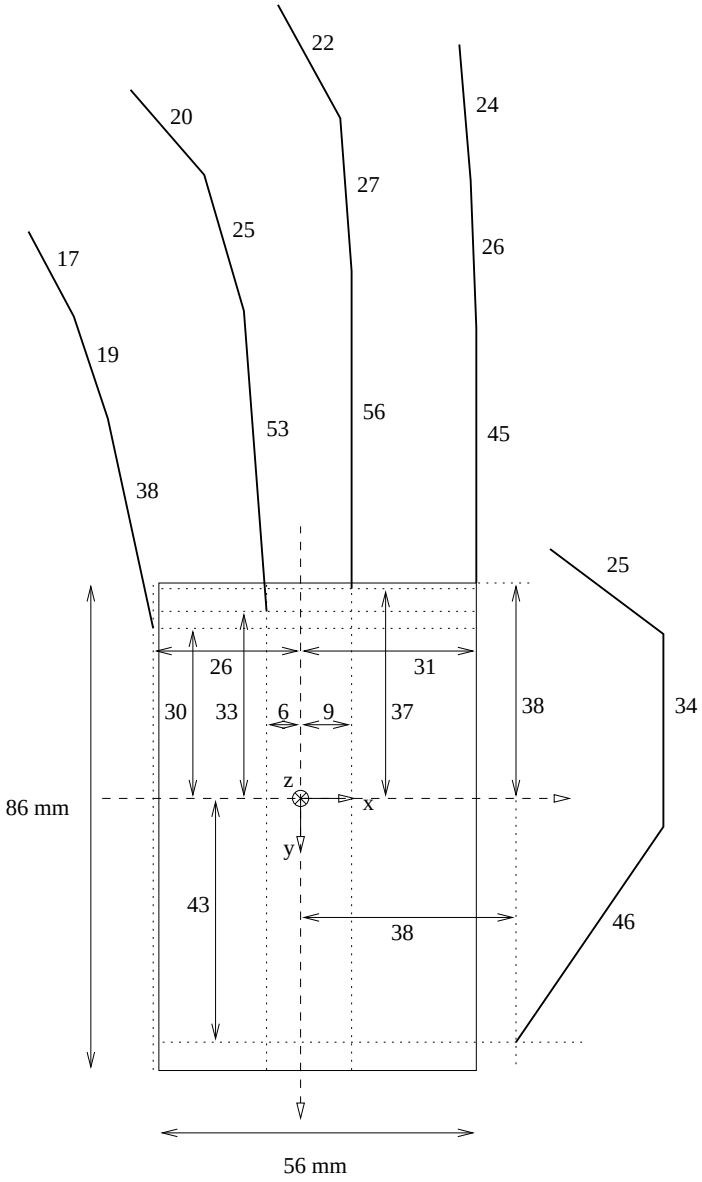
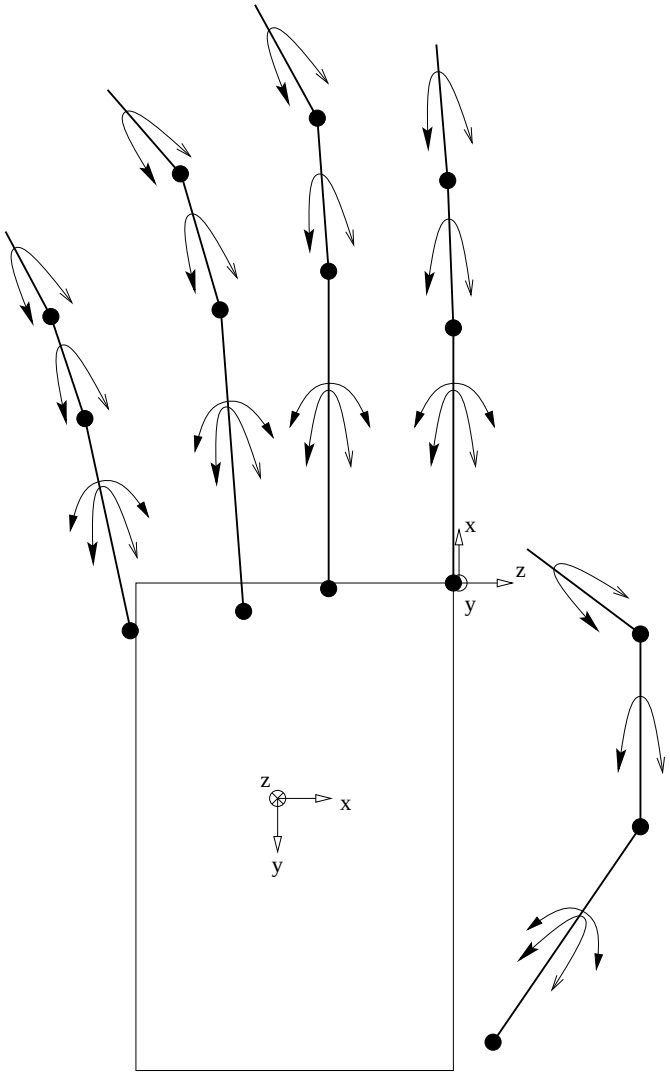
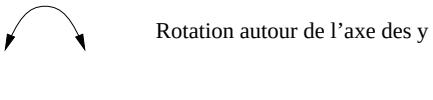
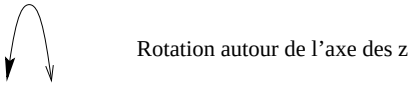


Figure 28 – *Forme du modèle 3D de la main (longueurs en mm)*







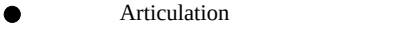


Figure 29 – *Mouvements du modèle 3D de la main*

Références

[BRO81] R.A. BROOKS, *Symbolic reasoning among 3D models and 2D images*, Artificial Intell. 17:285-348, 1981.

[BRO83] R.A. BROOKS, *Model-based three-dimensional interpretations of two-dimensional images*, IEEE Trans. Patt. Anal. Machine Intell. PAMI-5(2):140-150, 1983.

[CCD93] V. CASELLES, F. CATTE, T. COLL et F. DIBOS, *A geometric model for active contours*, Numerische Mathematik 66:1-31, 1993.

[CKS94] V. CASELLES, R. KIMMEL et G. SAPIRO, *Geodesic active contours*, Technical report, HP Labs, 1994.

[CKS95] V. CASELLES, R. KIMMEL et G. SAPIRO, *Geodesic active contours*, Proceedings of the 5th Proc. International Conference on Computer Vision, pp. 694-699, IEEE Computer Society Press, 1995.

[DER95] R. DERICHE et O. FAUGERAS, *Les EDP en traitement des images et vision par ordinateur*, Technical Report 2697, INRIA, 1995.

[FAU93] O. FAUGERAS, *Three-Dimensional Computer-Vision: a Geometric Viewpoint*, The MIT Press, 1993.

[GRI90] W.E.L. GRIMSON, *Object Recognition by Computer: The Role of Geometric Constraints*, MIT Press, 1990.

[HOG84] D.C. HOGG, *Interpreting Images of a Known Moving Object*, Thesis, University of Sussex, 1984.

[LEE95] J. LEE et T.L. KUNII, *Model-Based Analysis of Hand Posture*, Technical Report 95-2-002, DCH and DCS, University of Aizu, 1995.

[MSV93] R. MALLADI, J.A. SETHIAN et B.C. VEMURI, *A topology independant shape modeling scheme*, SPIE, 2031:246, 1993.

[MSV95] R. MALLADI, J.A. SETHIAN et B.C. VEMURI, *Shape modeling with front propagation: A level set approach*, PAMI, 17(2):158-175, 1995.

[PEN95] A. PENTLAND, *Machine Understanding of Human Action*, Technical Report 350, PCS, Media Lab, MIT, 1995.

[POP94] Arthur R. POPE, *Model-Based Object Recognition, A Survey of Recent Research*, Dept. of Computer Science, University of British Columbia, Technical Report 94-04, Janvier 1994.

[REH95] J.M. REHG (et T. KANADE), *Visual Analysis of High DOF Articulated Objects with Application to Hand Tracking*, These CMU-CS-95-138, School of Computer Science, Carnegie Mellon University, April 1995.

[SET95] J.A. SETHIAN, *Theory, algorithms and applications of level set methods for propagating interfaces*, Technical Report PAM-651, CPAM, University of California, Berkeley, 1995.

Remerciements:

Je tiens tout d'abord à remercier Olivier Faugeras, Directeur de Recherches à l'INRIA Sophia-Antipolis, pour m'avoir accueilli au sein de son projet Robotvis, ainsi que pour ses conseils.

Je remercie l'ensemble du projet Robotvis, dont Rachid Deriche, Thierry Vieville, Zhengyou Zhang, Luc Robert, Jean-Luc Szpyrka, Imed Zoghlami, Frederic Devernay, Pierre Kornprobst, Sylvain Bougnoux, Robert Stahr et Philippa Hook pour leur aide.

Je remercie Jim Rehg pour m'avoir aidé à construire le modèle 3D de la main.

Enfin, je remercie Renaud Keriven pour m'avoir fourni le code du programme d'évolution des contours géodésiques.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399